Practical Software Quality

A guide in progress

Presented at 2015 Flight Software Workshop by co-authors:
Dan Painter – NASA IV&V
Matt Rhodes - MathWorks

What's your [mis]fortune cookie say?

What is quality?

"The standard of something as measured against other things of a similar kind; the degree of excellence of something."

Oxford English Dictionary

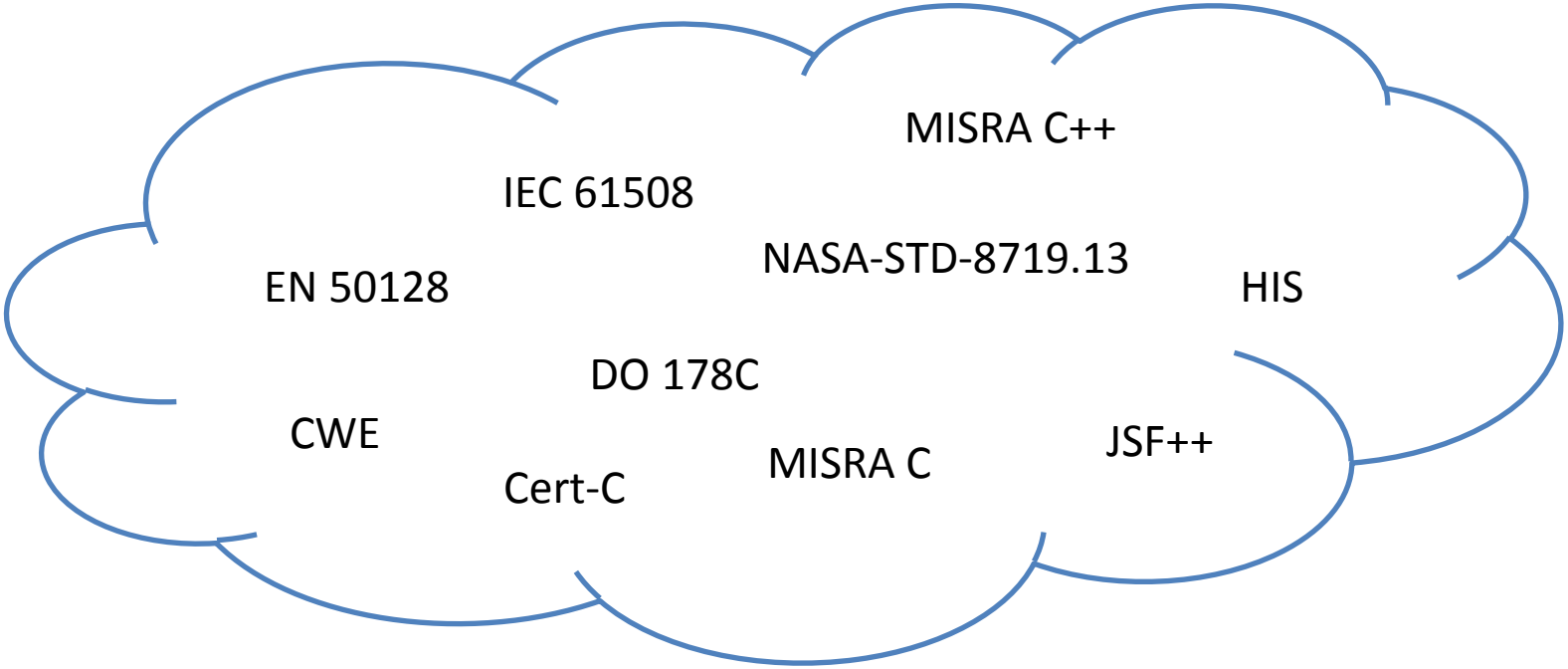
What is quality?

*"The **standard** of something as **measured** against other things of a similar kind; the degree of excellence of something."*

Oxford English Dictionary

Standards and measures

Standards can help you figure out what, and sometimes how, you should measure



Your software quality plan ***should*** help you figure out when to measure

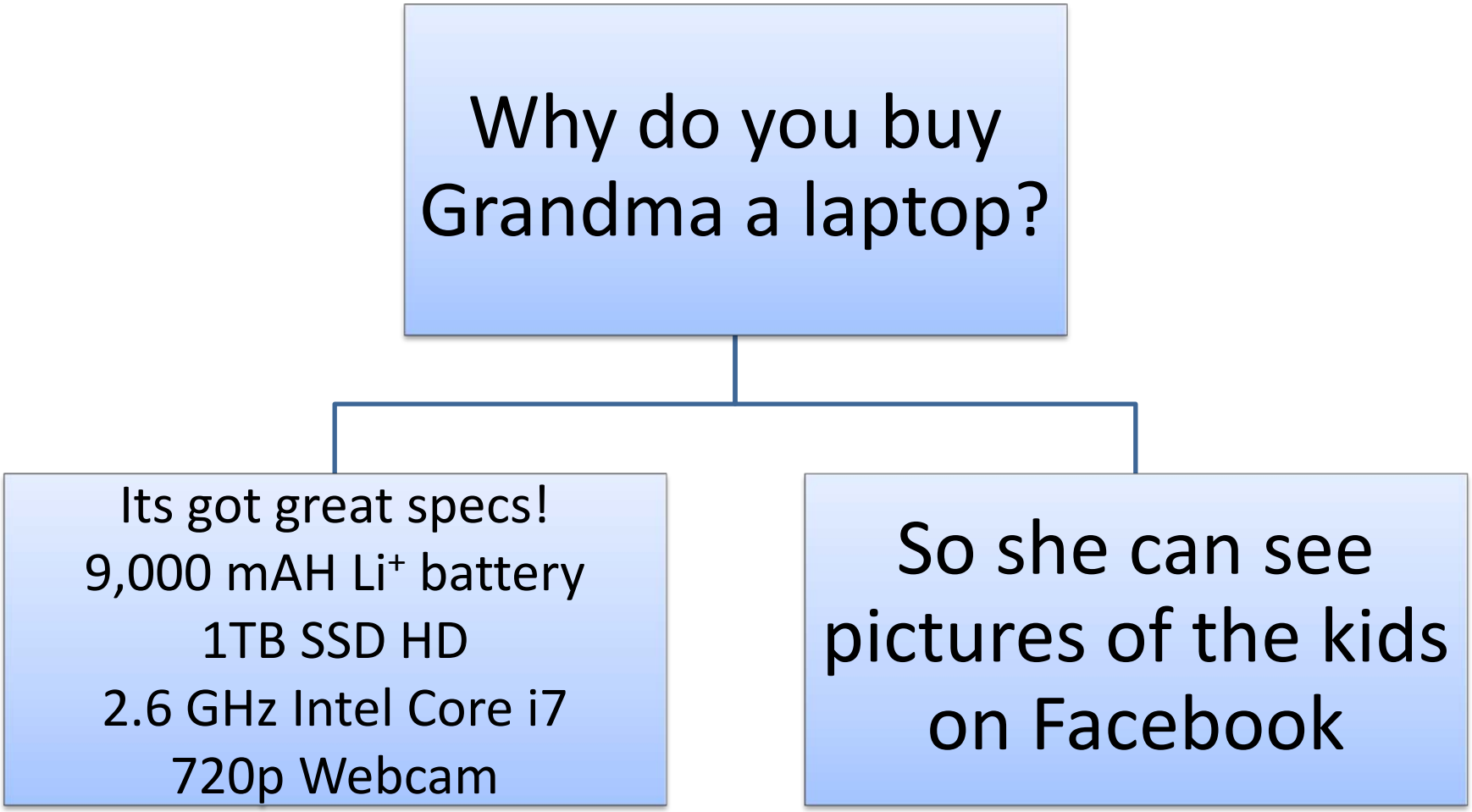
The impetus of software quality

The value of the software, from the customer's perspecrive, should drive the quality requirements.

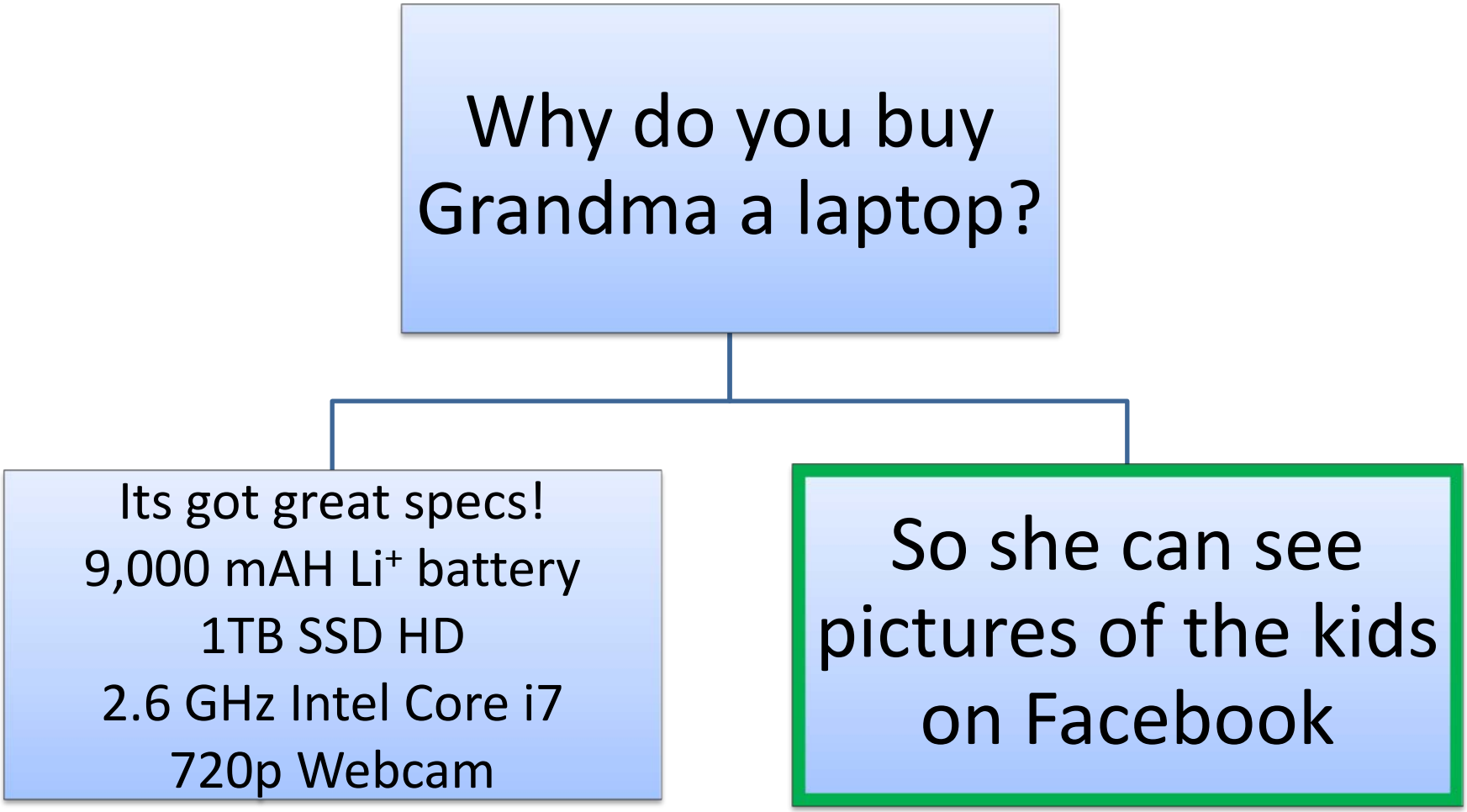
axiom

required software quality $\propto$ software value

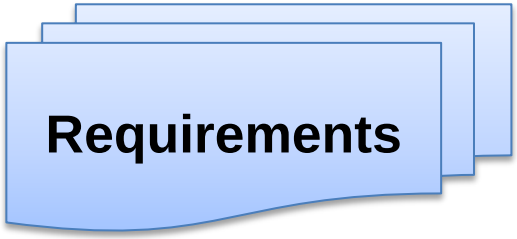
Understand the value



Understand the value

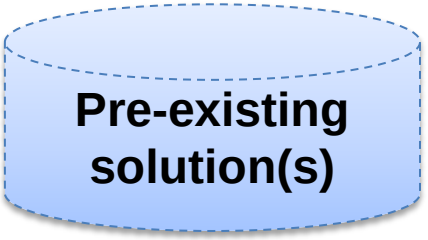


Specifications to enable value



- Customer driven requirements
- Industry driven process requirements
- Regulations
- Explicit quality requirements

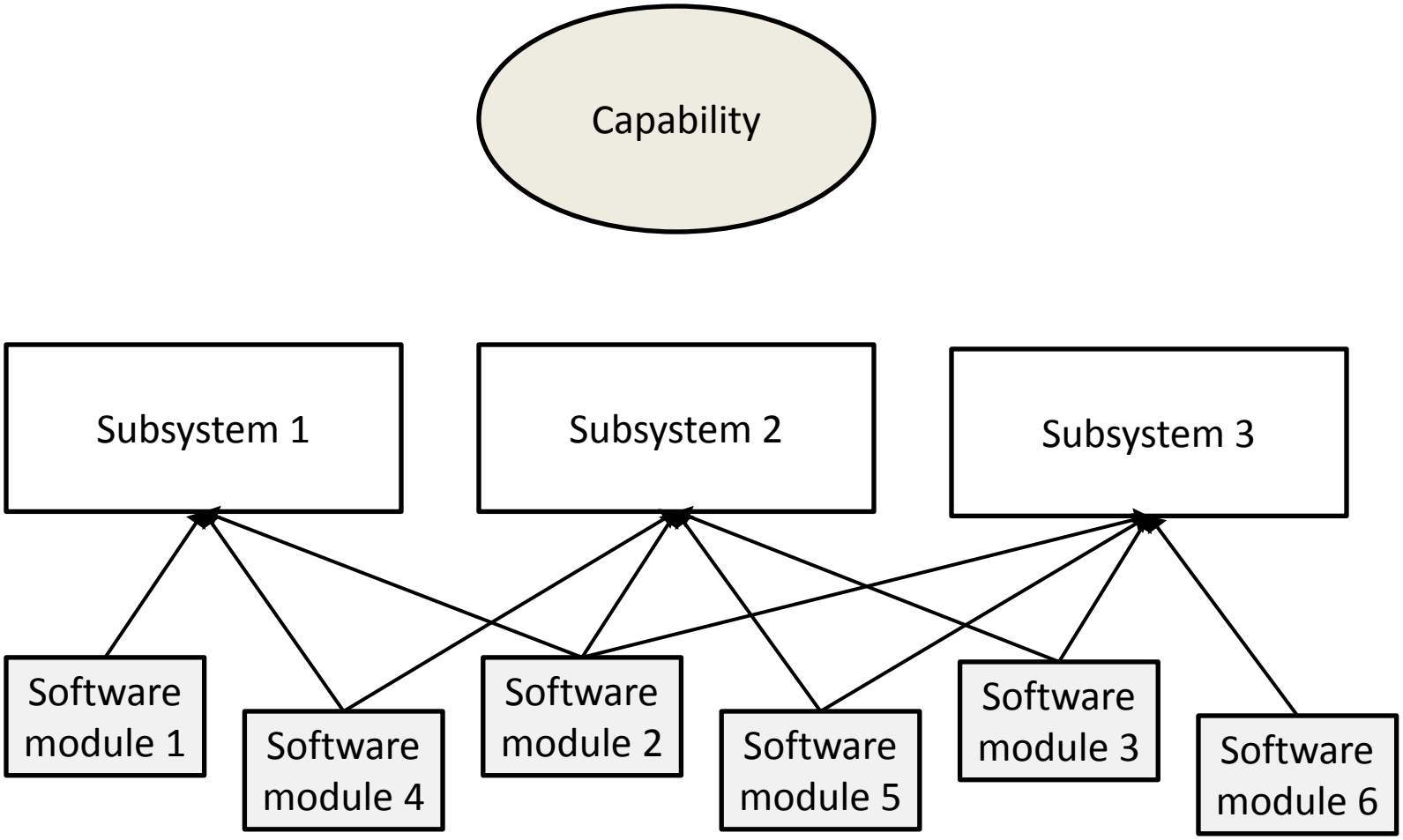
- Budget – Cost
- Budget – Time
- Budget – Resources



- Legacy software
- COTS sotware
- Libraries

Don’t overcomplicate; focus on what is important to the customer

Value is realized as capability



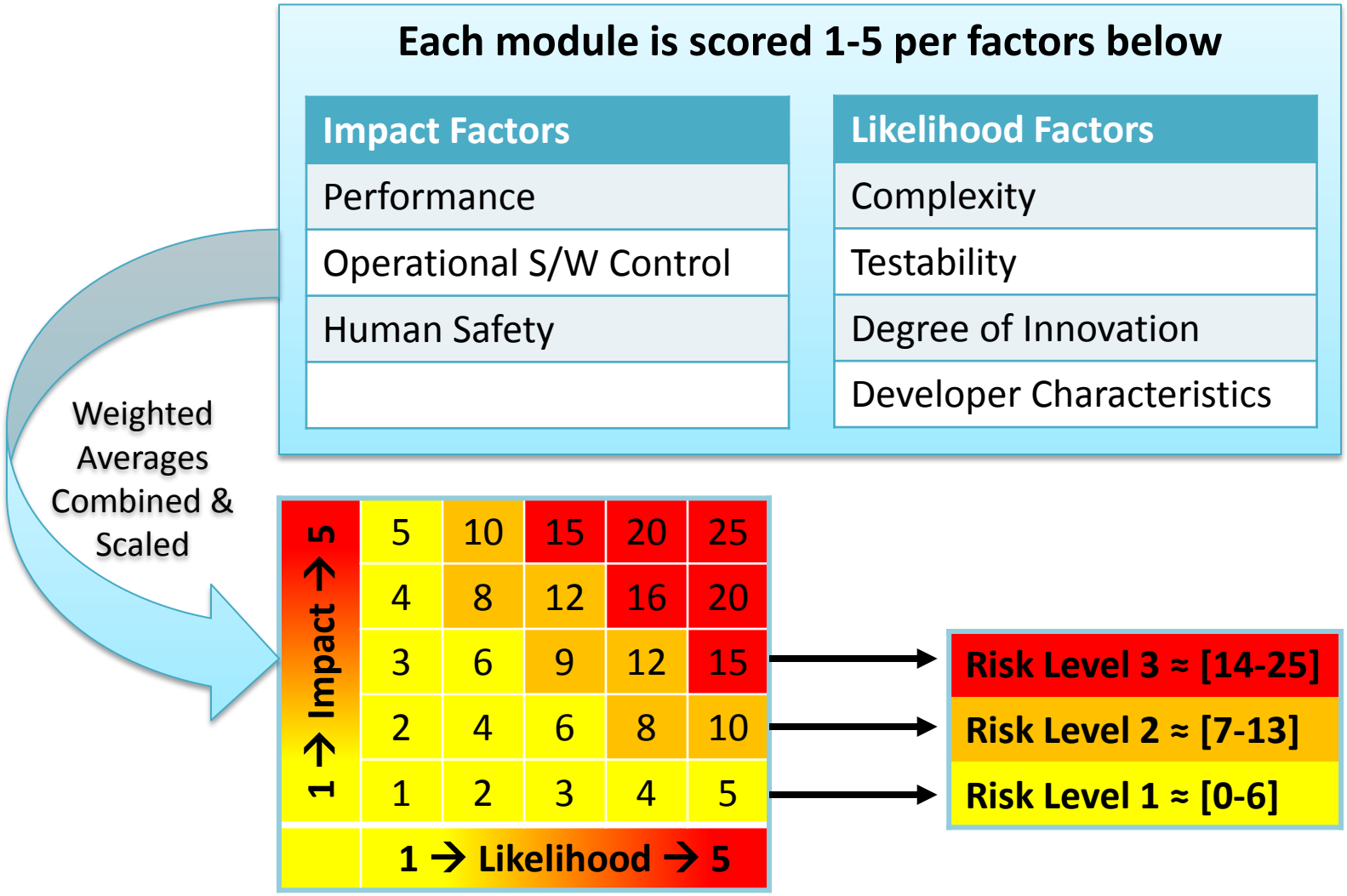
Protecting value from risk

Risks that could prevent the customer from realizing a desired capability indicate the value of the software itself is at risk.

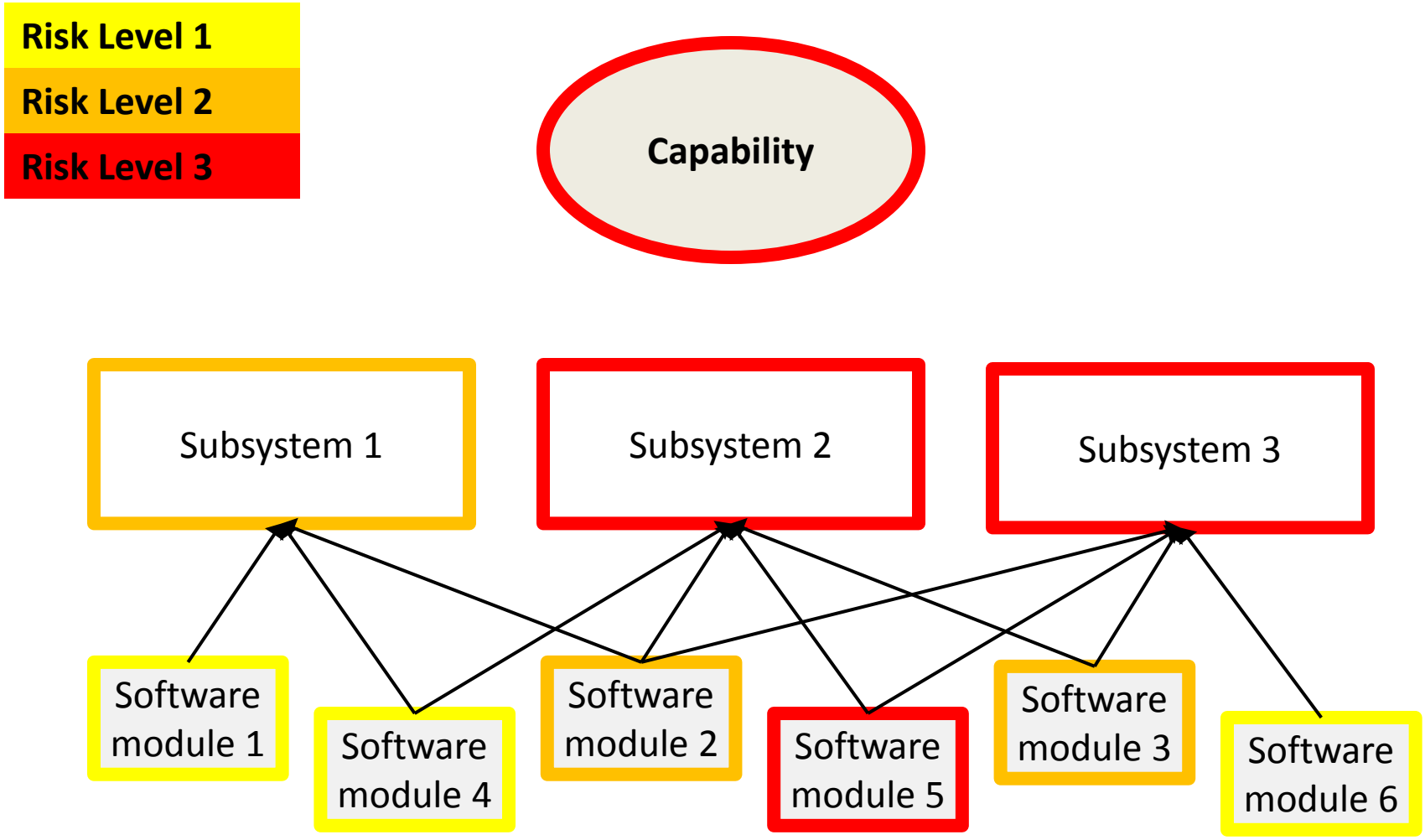
axiom

Unmitigated risks degrade delivered value

Per software module risk assessment



Risk flows up to capabilities



Capability	Software Component	Software Modules
Launch to Orbit	None	None
Approach to Target	Trajectory Control	Math
		GNC
		Attitude Control
		Navigation
	Attitude Control	Math
		GNC
		Attitude Control
		Navigation
Maintain Flight Systems	Establish and Maintain Power	Battery
		Power
	Establish and Maintain Thermal Control	Thermal
	Perform Fault Detection	FSW
	Establish and Maintain Communication	Telecom
		Downlink
		Uplink
		Command
		Telemetry

Leverage the progress of others

There is no reason to repeat the same mistake; make sure to mitigate known risks with proven risk mitigation techniques.

axiom

Start with known mitigations for known risks

Known risks: Enemies of Quality

Unrealistic expectations

Incorrect specification

Bad coding practices and constructs

Inadequate testing

Example risk mitigation activities

Building the right thing

- Prototyping
- Simulation
- Requirements tracing
- Design reviews

Building the thing well

- Code generation
- Unit testing
- Coding standards
- Code reviews
- Monte Carlo testing

Confirmation of building the right thing well

- Static analysis
- Integration testing
- Coverage testing
- Code metrics

Other Risk Mitigations

- Independent verification and validation
- Experience and training
- Continuous quality plan re-evaluation

Each of these mitigation activities combats one or more of enemies of quality.

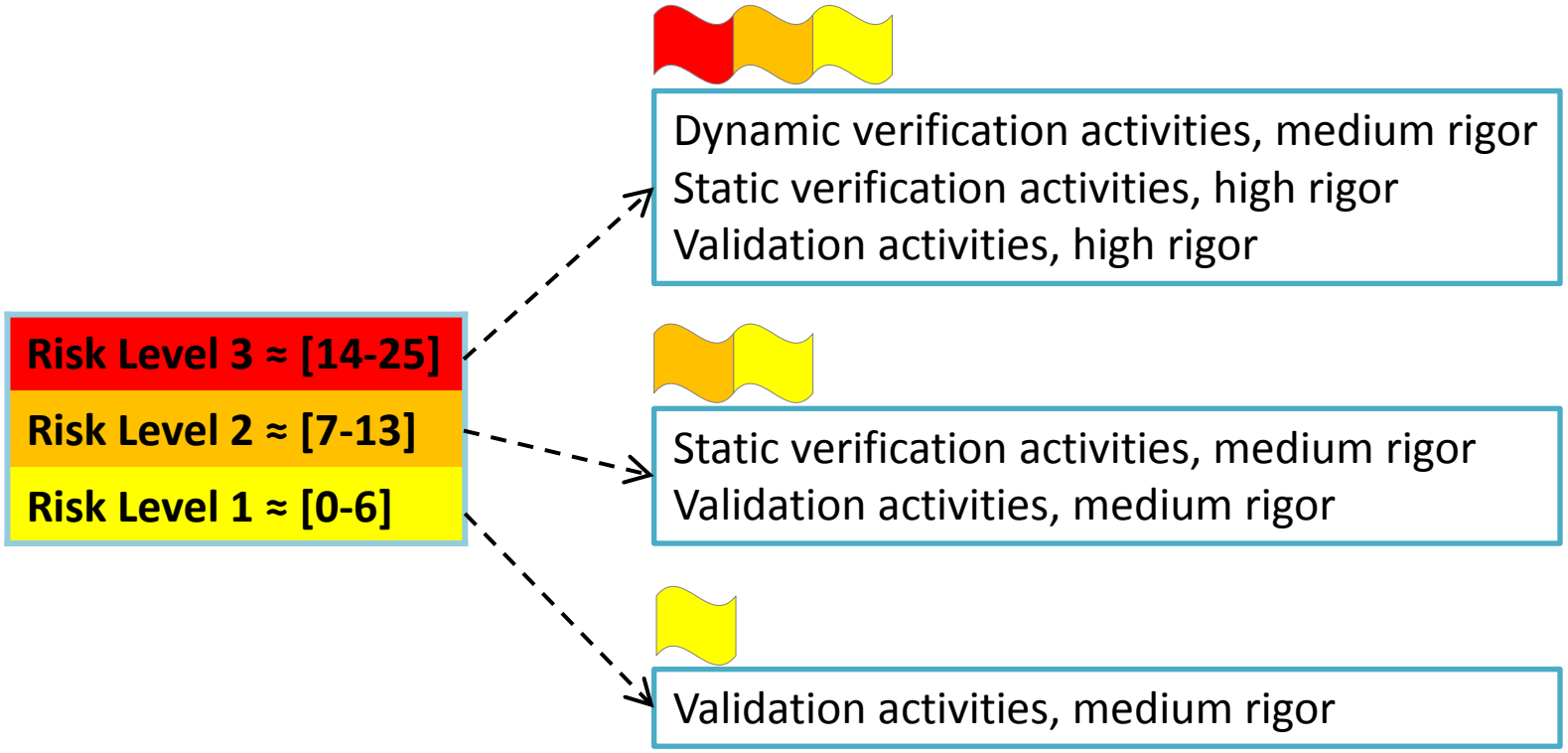
Scope with the risk assesment

Since exhaustive testing is out of reach, risk mitigation activities should be scoped relative to the determined risk levels.

axiom

Risk mitigation $\propto$ risk level

Example risk mitigation scoping

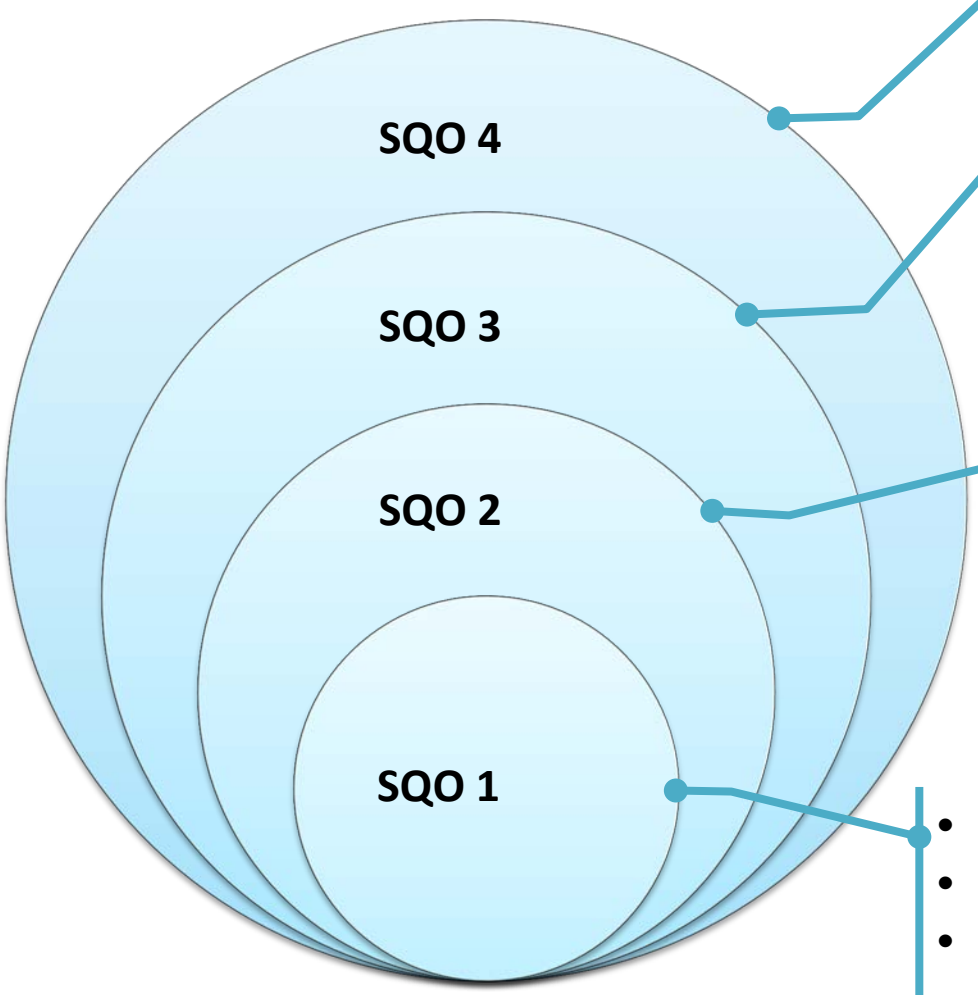


Each risk level dictates risk mitigation methods applied with specific levels of rigor.

An analogy for rigor

How = **Rigor** of strategy application ≈ **How many stones do I need to look under?**

Rigor applied to static code analysis



- Prove code safe

- Analyze third set of potential run-time errors
- Apply coding standard X checks

- Analyze unreachable branches
- Analyze second set of potential run-time errors
- Identify external interfaces and perform tainted data analysis

- Identify systematic run-time errors
- Analyze non-terminating constructs
- Analyze first set of potential run-time errors

Rigor applied to static code analysis

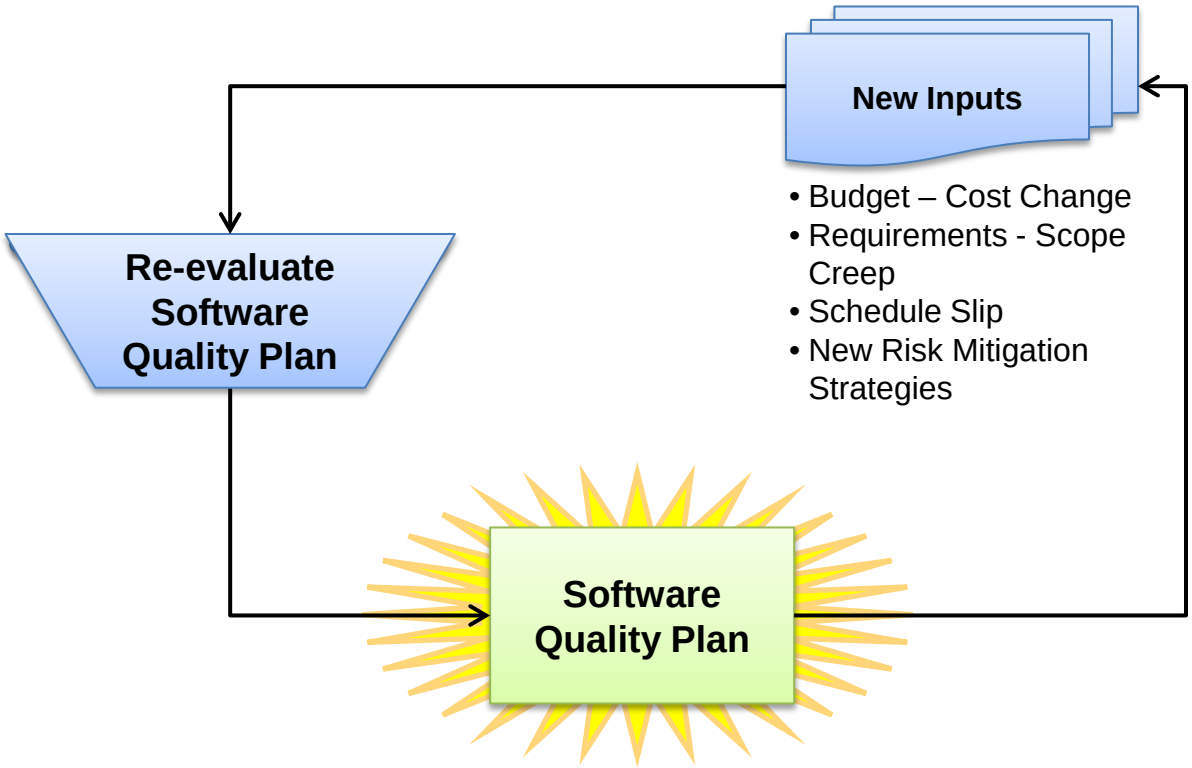
Assurance Task	SQO Level 1	SQO Level 2	SQO Level 3	SQO Level 4	SQO Level 5	SQO Level 6
Develop Quality Plan (AT-1)	X	X	X	X	X	X
Identify Software Build Information (AT-2)	X	X	X	X	X	X
Identify Source Code Metrics (AT-3)	X	X	X	X	X	X
Apply Standards Based Rules (AT-4)	OPTIONAL per IV&V effort	OPTIONAL per IV&V effort	OPTIONAL per IV&V effort	OPTIONAL per IV&V effort	OPTIONAL per IV&V effort	OPTIONAL per IV&V effort
Identify Systematic Runtime Errors (AT-5)		X	X	X	X	X
Analyze Non Terminating Constructs (AT-6)		X	X	X	X	X
Analyze Unreachable Branches (AT-7)			X	X	X	X
Identify External Interfaces (AT-8)				X	X	X
Analyze First Subset of Potential Runtime Errors (AT-9)				X	X	X
Analyze Second Subset of Potential Runtime Errors (AT-10)					X	X
Analyze Third Subset of Potential Runtime Errors (AT-11)						X
Prove Code Safe (AT-12)						X (targeted modules)
Perform Tainted Data Analysis (AT-13)					OPTIONAL or Required for Information Assurance/Security Focused Analysis	OPTIONAL or Required for Information Assurance/Security Focused Analysis
Perform Dataflow Analysis (AT-14)						X

Rigor applied to static code analysis

Impact List	Impact Definition	Impact Level	SQO Level
MISRA or standards compliance	Neither causes harm to the system nor a programmer mistake. These are simply good practices or generally accepted standards to follow.	Trivial	1,2,3,4,5,6
Deadlock	Two or more threads are waiting for each other to finish causing the process to freeze. These are related to Semaphores, Mutexes, and Race conditions.	Critical	2,3,4,5,6
Memory leak	Improper memory management. These involve improper or neglected deallocation or use of memory.	Minor - Major	2,3,4,5,6
System crash	Impacts system/crew safety which could lead to Loss of vehicle, loss of mission, or loss of life.	Critical	2,3,4,5,6
Undefined behavior	Code defects whose behavior is not specified under certain conditions. The behavior may vary depending on the implementation, environment, or semantics. Resulting behavior can range from benign to critical.	Major - Critical	2,3,4,5,6
Possible programmer mistake	Does not cause any major or critical issues, but areas in code that may be worth a look to determine if code was intentional or not.	Minor	3,4,5,6
Unexpected behavior or results	Suspicious code that may negatively affect the behavior, code flow, or calculation result if the code was not programmed as intended.	Minor - Major	3,4,5,6
Unreachable code	Written code that will not be executed. These could either be commented out code or defensive code. Worth investigating to see if intentional.	Minor - Major	3,4,5,6
Data loss	Chance to truncate data when assigning between objects, storing results of a calculation, or passing data as arguments, when the new storage type is smaller.	Major	4,5,6
Data exposure	Security vulnerability allowing supposedly inaccessible or private data to be modified by a malicious user.	Minor - Major	5,6
Security	Security vulnerabilities that do not overlap with another impact category. These include the use of unsafe functions, unverified or tainted inputs, or weaknesses prone to user exploitation.	Minor - Major	5,6
Code cleanliness	Good practices to observe near the completion of a project such as declaring objects as const or non-const when appropriate.	Trivial	6
Performance	Impacts system performance such as timing or memory usage.	Minor	6
Portability or cause compile issues	Code defects that may not be an issue on the current system but may not work if compiled on a different environment or if implementation was not well understood.	Trivial	6
Readability and maintainability	No impact on the system other than the possibility of confusion if code was shared/maintained by multiple developers or reused in another project without proper rationale included.	Trivial	6
Redefined behavior	Built-in commands or operators are overloaded or redefined to have new behavior. May cause confusion, however it is a non-issue if the system is well understood and documented.	Trivial - Minor	6
Unused data	Possible development oversight. A parameter, status or calculation result was not used, indicating there may have been an initial intent but forgotten.	Minor	6

Putting it all together

Software quality plan reassesment



Your quality plan is an evolving, living process.

What can IV&V provide?

Assurance

- Code analysis
- Simulation
- Proven evidence based approach

Cost savings

- Subject matter experts

Confidence

- Safe
- Error Free
- Meet your needs

What can MathWorks provide?

Tools

- MATLAB, Simulink, Polyspace and more

Expertise

- Consulting
- Training
- Process assessment
- Model Based Design guidance

Community

- File Exchange
- MATLAB Answers
- Blogs

Contribute or ask questions